logic is carried out by a first work team and the development of the data access functionality is carried out by a second team. To facilitate simultaneous development by the two teams, Bowman-Amuah defines a data state class, which defines the basic data types for the raw data to be interchanged between the components developed by the two teams and serves as a "contract" between them. *See e.g.*, abstract, summary of invention, claims, and col. 288, line 46 – col. 289, line 23. In sum, Applicants' invention relates to datastore encapsulation whereas Bowman-Amuah relates to separation of work responsibilities during computing development.

Upon appreciation of the "big picture" difference between Applicants' claimed invention and Bowman-Amuah, it should also become evident that the individual elements of Bowman-Amuah relied upon by the Examiner are also quite different from the elements recited in independent claim 1, specifically:

> A framework for isolating a business component form specific implementations of a datastore – The text relied upon by the Examiner appears to be randomly selected based on key words such as "business component" (see col. 3, lines 66-67) and "database" (see col. 8, lines 58-60), as the text does not provide a uniform theme or teaching related to datastore encapsulation. As explained previously, the abstract relates to the separation of work responsibilities during computing development with is not the same or equivalent to Applicants' preamble reciting datastore encapsulation.
>
> (a) a database wrapper in communication with a business component – The text at col. 143, lines 19-21, relied upon by the Examiner states that "at the architecture level, wrappers often provide database interface objects to shield the application from the database vendor." While a true statement regarding the general functionality of wrappers, this statement is not provided in the context of a specific framework as claimed by Applicants. Respectfully, this statement is akin to "a car provides transportation" – while true, it provides little detail about the specifics of the layout of the car (or in this case, the claimed framework). The Examiner next jumps over 70 columns of text to col. 216, lines 13-18, which relates to the use of a

wrapper to access a legacy system, which typically means that an old computing system such as a mainframe is dressed up with a new interface such that the old system can communicate with newer systems. These two isolated, unrelated passages do not teach or suggest the recited element of a database wrapper in communication with a business component in the overall context of datastore encapsulation.

(b) a domain object factory in communication with the database wrapper – Jumping another 40 columns, the text relied upon by the Examiner at col. 259, lines 11-17, relates to "garbage collection," that is, the clean up of orphaned server context that remains after a session with a client ends, which has little or no relationship to datastore encapsulation. The lack of overall relevance greatly impairs the usefulness of the text, if any. Furthermore, the Examiner specifically relies upon the phrase "constructing domain objects," but the reference is silent as to how such domain objects are constructed, and thus provides no teaching as to the use of a domain object factory. In sum, this isolated passage related to garage collection does not teach or suggest the recited element of a domain object factory in communication with the database wrapper in the overall context of datastore encapsulation.

(c) a domain object in communication with the domain object factory – The text and figures relied upon by the Examiner make no mention of a domain object factory, but rather refer to a domain object 16100 in communication with a database via a data handler 16102, which serves as a communication link between the domain object 16100 and the database. In other words, the domain object 16100 requires a separate component (i.e., the data handler) to access data from the database. In contrast, Applicants, through the use of a domain object factory, instantiate a domain object that is specific to each underlying datastore (see page 8, line 14 – page 9, line 12), thereby allowing the domain object to access information directly from the datastore and

avoiding the need for a "middleman" component such as data handler 16102.

(d) a datastore in communication with the domain object – Figure 159 appears to show a domain object in communication with a datastore, but again this disclosure is taken in isolation completely devoid of any meaningful context as an anticipating disclosure. More specifically, Fig. 159 does not show the relationship between a domain object generated by a domain object factory in communication with an underlying datastore as recited in Applicants' claims.

In sum, there is no common embodiment or thread linking the various and widespread patchwork of text relied upon by the Examiner, which suggests that the Examiner appears to be impermissibly picking and choosing across the reference rather than reading the teaching of Bowman-Amuah as a whole. When read as a whole, Bowman-Amuah clearly relates to a completely separate computing problem and solution, namely, the problems encountered via bifurcation of development between a business logic team and a data access team. Therefore, Bowman-Amuah does not disclose datastore encapsulation as disclosed and claimed by Applicants, and thus independent claim 1 and claims 2-4 depending therefrom cannot be anticipated by Bowman-Amuah.

With respect to independent claim 9, much of the text relied upon by the Examiner is the same as that distinguished previously with respect to claim 1, and thus for the sake of brevity, Applicants will not repeat the distinctions but make reliance upon the same. The newly relied upon sections of text are differentiated as follows:

(b) implementing the database wrapper – The Examiner's reliance upon the text at col. 127, line 60 - col. 128, line 4, is puzzling, as this passage has nothing to do with a database wrapper or implementation thereof. Skipping nearly 100 columns, the Examiner's reliance upon col. 232, lines 9-36, and related Fig. 104 appears to be based upon the presence of the terms "wrapper" and "implemented," but appears to have no further

relationship to the implementation of a database wrapper as recited by Applicants.

(d) implementing the domain object factory – As noted above, Bowman-Amuah does not disclose a domain object factory, and therefore cannot further disclose the implementation of a non-disclosed element. The text at col. 124, lines 15-19, relied upon by the Examiner is related to object-oriented methodologies whereby a system is modeled by breaking (*i.e.*, decomposing) the system into component parts such as domain objects and processes. This general statement regarding decomposition methodology does not teach or suggest the recited element of implementing the domain object factory in the overall context of datastore encapsulation.

(f) implementing the domain object to retrieve data from a datastore – Jumping over 100 columns, the text at col. 245, lines 17-21, appears to disclose instantiation of a domain object to retrieve data from a datastore, but again this disclosure is taken in isolation completely devoid of any meaningful context as an anticipating disclosure. More specifically, the text does not show the relationship between a domain object generated by a domain object factory being implemented to retrieve data from a datastore as recited in Applicants' claims.

In sum, Bowman-Amuah does not disclose the datastore encapsulation method as disclosed and claimed by Applicants, and thus independent claim 9 cannot be anticipated by Bowman-Amuah.

§ 103 Rejections

Claims 5-8 and 10-14 stand rejected under 35 U.S.C. § 103(a) as being unpatentable over Bowman-Amuah (U.S. Pat. 6,442,748) in view of Brownell (U.S. Pat. 6,009,266). Applicants note that as a formality, Brownell does not appear on a copy of the notice of references cited (form PTO-892) received with the 06/20/02 or 12/06/02 office actions. Applicants respectfully submit that the combination of Bowman-Amuah and Brownell does not establish a *prima facie*

case of obviousness as to claims 5-8 and 10-13. According to MPEP § 2142, three basic criteria must be met to establish a *prima facie* case of obviousness:

> First, there must be some suggestion or motivation, either in the references themselves or in the knowledge generally available to one of ordinary skill in the art, to modify the reference or to combine reference teachings. Second, there must be a reasonable expectation of success. Finally, the prior art reference (or references when combined) must teach or suggest all the claim limitations. The teaching or suggestion to make the claimed combination and the reasonable expectation of success must both be found in the prior art, and not based on applicant's disclosure.

Assuming for sake of argument that the combination of Bowman-Amuah and Brownell is proper (without conceding same), the Examiner has nonetheless failed to establish a *prima facie* case of obviousness as such a combination does not teach or suggest all of the claim limitations. Claims 5-8 and 10-13 depend from and incorporate the limitations of independent claims 1 and 9, respectively. As discussed previously, Bowman-Amuah does not disclose each and every element of independent claims 1 and 9 and, more specifically, does not disclose several recited elements related to datastore encapsulation. Brownell is not cited by the Examiner for the purpose of providing these missing datastore encapsulation elements, and in any event does not do so even if relied upon for such. In sum, a *prima facie* case of obviousness has not been established as to claims 5-8 and 10-13.

Independent claim 14 was rejected under 35 U.S.C. § 103(a) as being unpatentable over Bowman-Amuah (U.S. Pat. 6,442,748) in view of Brownell (U.S. Pat. 6,009,266). Independent claim 14 is likewise drawn to method for encapsulating a database. Again, the combination of Bowman-Amuah and Brownell (assuming without conceding such to be proper) does not disclose each and every element of independent claim 14 and, in particular, does not disclose the numerous elements reciting database encapsulation functionality. With respect to independent claim 14, much of the text relied upon by the Examiner is the same as that differentiated previously with respect to independent claims 1 and 9, and thus for the sake of brevity, Applicants will not repeat the distinctions but make reliance upon the same. The newly relied upon sections of text are differentiated as follows:

(b) using the database wrapper to begin a database session;

(f) ending the database session; and

(g) returning the domain object to the business component – The text at col. 69, lines 39-58, relied upon by the Examiner is a general disclosure regarding the functionality of message-oriented middleware (MOM), which includes opening communications sessions and transferring data therein. However, this general statement regarding messaging functionality does not teach or suggest the recited elements in the overall context of the claimed datastore encapsulation method.

(e) converting the domain object from a persistent state to a transient state – The text at col. 11, lines 1-17, of Brownell appears to show the conversion of an object from a persistent state to a transient state, but not in the specific context of the claimed datastore encapsulation method. More specifically, the text does not show the conversion of a domain object generated by a domain object factory as recited in Applicants' claims. Furthermore, given the numerous deficiencies of the primary reference (i.e., Bowman-Amuah) as described previously, Applicants respectfully submit that the secondary reference (*i.e.*, Brownell) cannot fairly be read to correct such deficiencies, nor could a reasonable argument be made for their combination. In other words, it is difficult to see how one would be motivated to modify Bowman-Amuah with the teaching of Brownell when Bowman-Amuah does not come close to describing Applicants' claimed datastore encapsulation framework and method.

In sum, Bowman-Amuah does not make obvious each and every element of independent claim 14, nor does Brownell make up for the lack of teaching in Bowman-Amuah.

Conclusion

Applicants respectfully submit that the present application is in condition for allowance. If the Examiner has any questions or comments or otherwise feels it would be helpful in expediting the application, he is encouraged to telephone the undersigned at (972) 731-2288.

The Commissioner is hereby authorized to charge payment of any fees associated with any of the foregoing papers submitted herewith to Deposit Account No. 21-0765, Sprint.

Respectfully submitted,

CONLEY ROSE, P.C.

Date: 2-19-03

Rodney B. Carroll
Reg. No. 39,624

ATTORNEY FOR APPLICANTS

5700 Granite Parkway, Suite 330
Plano, Texas 75024
Telephone: (972) 731-2288
Facsimile: (972) 731-2289